Separability versus prototypicality in handwritten word-image retrieval

van Oosten, Jean-Paul; Schomaker, Lambertus

Published in:
Pattern recognition

DOI:
10.1016/j.patcog.2013.09.006

Publication date:
2014

Separability versus Prototypicality in Handwritten Word-image Retrieval

Jean-Paul van Oosten*, Lambert Schomaker

Dept. of Artificial Intelligence, University of Groningen, The Netherlands

Abstract

Hit lists are at the core of retrieval systems. The top ranks are important, especially if user feedback is used to train the system. Analysis of hit lists revealed counter-intuitive instances in the top ranks for good classifiers. In this study, we propose that two functions need to be optimised: (a) In order to reduce a massive set of instances to a likely subset among ten thousand or more classes, separability is required. However, the results need to be intuitive after ranking, reflecting (b) the prototypicality of instances. By optimising these requirements sequentially, the number of distracting images is strongly reduced, followed by nearest-centroid based instance ranking that retains an intuitive (low-edit distance) ranking. We show that in handwritten word-image retrieval, precision improvements of up to 35 percentage points can be achieved, yielding up to 100% top hit precision and 99% top-7 precision in data sets with 84 000 instances, while maintaining high recall performances. The method is conveniently implemented in a massive scale, continuously trainable retrieval engine, Monk.

Keywords: image retrieval, handwriting recognition, nearest centroid, support-vector machines, separability, prototypicality, historical manuscripts, big data, continuous machine learning

1. Introduction

In handwriting recognition, classification is often performed using statistical methods [1, 2]. The class indexed i with the highest posterior probability given the sample to be classified is chosen as the result of the classifier:

$$\text{Hypothesis}_X = \underset{i}{\text{argmax}}\, P(C_i|X) \quad \text{where } i \in \{1, N_{\text{classes}}\} \tag{1}$$

However, when the goal is word search, rather than automatic text transcription, the user is more interested in retrieval of word instances. Instead of

*Corresponding author

Email addresses: J.P.van.Oosten@ai.rug.nl (Jean-Paul van Oosten), L.Schomaker@ai.rug.nl (Lambert Schomaker)

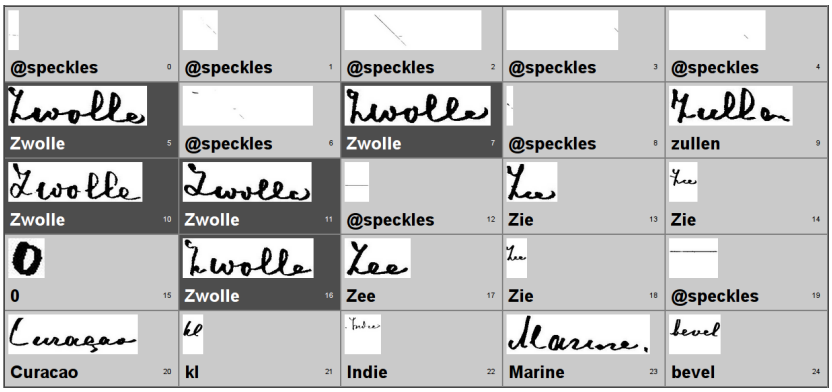


Figure 1: First 25 instances in a hit list of the word 'Zwolle'. Original test set performance: Accuracy: 99.2%, precision: 97.6% and recall: 97.6%. Note the faulty instances in the top ranks, upper row. In a realistic test condition with 12k distractors, actual precision is as low as 2.8%.

a single classification, the result is a sorted hit list H. Each instance indexed j is ranked with respect to the prototype or class-model corresponding to the search term:

$$H = \underset{j}{\text{sort}}(P(X_j|C)) \quad \text{where } j \in \{1, N_{\text{examples}}\} \tag{2}$$

Retrieval is usually performed on a large collection of instances, and only the top of the sorted list, representing the best ranking instances, is considered as interesting. Under such a condition, a large number of classes and a massive data collection can pose a problem, since for each query there is a large number of distractors, i.e., concerning instances from all classes, other than the target class.

This becomes apparent in retrieval engines for handwritten words in historical collections [3]. In the *Monk* system, twenty books of ≈1000 pages each contain millions of word zones or word candidates, and the lexicon is in the order of tens of thousand word class models. From the tradition of handwriting-recognition research, it seems reasonable to start with the classification problem (Eq. 1), using good shape features and a powerful classifier, such as, e.g., hidden-Markov models [4, 5] or the support-vector machine [6, 7]. For a word-mining task, such a classifier may be trained to discriminate a particular word class, and a ranked word list may be constructed, e.g., using the signed SVM discriminant value d_{SVM} for sorting. The basic assumption then is, that the distance from the margin, i.e., from the instances in the distractor classes, will be a good criterion for constructing a ranked hit list for a target class. However, upon applying this approach, we observed an interesting phenomenon in the resulting hit lists. As an example, Figure 1 shows the top-25 instances in a hit list for the word 'Zwolle'. The performance for the word classifier on the entire training set was 100% accuracy, with a 97% accuracy on an independent test

set ($k = 7$ folds, $\sigma = \pm 1\%$). Following regular testing procedures for SVMs, the training and the test sets were of similar size, each containing a quarter of positive examples (typically 50) and three quarters of negative or distractor examples. However, the resulting hit list contains a number of counter-intuitive samples (e.g., speckle images) in the early ranks, followed by a strand of correct classifications which is followed by a transitional stage of occasional errors.

The impression that a problem exists is confirmed by a larger-scale analysis of the results (Table 1), also using a realistic large set containing $\approx 12 \times 10^3$ distracting word instances in the test set. The results for *accuracy* and *recall* on the realistic data set confirm the hopeful expectancies which were raised by the regular training and test sets. However, the *precision* of the output drops abysmally, to about 1% in the worst cases, notably for the classes with a limited number of training examples (Table 1, lower right). It should also be noted that a number of 12K distractors (1/1200) is much more realistic than a 1/4 rule which is commonly accepted in academic testing.

It is clear that something is needed to improve on the performance. User appreciation of hit lists is of paramount importance in live and continuously trainable systems that rely on user annotation over the internet, such as *Monk*[3, 8]. Figure 2 shows how hit lists are used in the Monk system. Upon giving the first handful of (bootstrap) examples, a usable machine-learning system should be able to produce an acceptable ranking such that newly found instances of the same class can be easily labelled. The above, concrete observation thus gives rise to a more fundamental question: How is it possible that accuracy is not a good predictor of precision in a retrieval context?

In this study, we will 1) analyse the reason for unexpected, low precision in

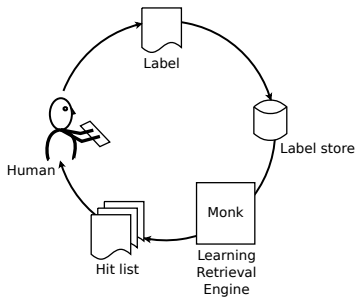


Figure 2: Schematic overview of how users utilise the hit lists to label new word images in a continuously learning retrieval engine (Monk). A hit list is presented to the user, who produces a label for an unlabelled word. This label is stored in the label store, which is then processed by the retrieval engine to produce a new hit list. The interface facilitates the quick labelling of a large number of instances that match the query word.

Table 1: Counter-intuitive, low precision results for good classifiers

		Accuracy		Recall		Precision	
Set	N_{examples}	Mean	σ	Mean	σ	Mean	σ
Test	120+	0.98	0.02	0.97	0.05	0.96	0.07
	60-120	0.97	0.03	0.95	0.10	0.91	0.13
	35-60	0.97	0.04	0.93	0.15	0.85	0.19
	7-35	0.96	0.04	0.68	0.42	0.57	0.40
+12K Distractors	120+	0.99	0.01	0.97	0.05	0.26	0.26
	60-120	0.98	0.02	0.95	0.10	0.06	0.12
	35-60	0.97	0.02	0.93	0.15	0.03	0.06
	7-35	0.97	0.04	0.68	0.42	**0.01**	0.05

presumably well-performing classifiers; 2) explore a number of methods to counteract the precision drop and 3) present a convenient approach using nearest-centroid matching, with results in a similar ballpark as the abovementioned SVM approach, at the same time however, avoiding expensive training on the tens of thousands of word classes.

2. Separability versus Prototypicality

Problem: The SVM is a discriminative classifier, optimised for *classification* (Eq. 1). The class of an unknown sample X (Figure 3) is decided by determining on which side of the decision boundary β the sample falls. For *retrieval* purposes, it appears reasonable to use the distance to the boundary, $d(X, \beta)$, as a ranking measure: the farther the instance is located from the boundary, the more certain an SVM classifier is of the classification.

Unfortunately, this gives unexpected results, such as shown in Figure 1 for the query word 'Zwolle'. Instances that are ranked at the top (`@speckles`) appear to be counter intuitive to a human user. It seems that there are two problems: 1) the distance to the boundary is not an intuitive measure, and 2) a fairly large number of distractors causes noise in a hit list, and consequently, a lower precision. The implication is that enlarging the dataset increases the probability that incorrect instances occur even before the first correct hit. This has a large impact on the user appreciation and is hard to explain. More informally: Many hits do not appear similar to the user's expected, canonical prototype for the query.

Proposed explanation: In order to give a plausible explanation of this phenomenon, we present a schematic, two-dimensional overview. The position of an instance X in Figure 3 has a large distance $d(X, \beta)$ from the boundary β (which is desirable). However, the instance X is not very prototypical, being located far from the known instances of the target class A. In other words, the distance of the instance X to the prototype, or centroid of class A, $d(X, \lambda_A)$, is large.

The support-vector machine training mechanism has an emphasis on *separability*: the ability to categorise and separate class instances from non-

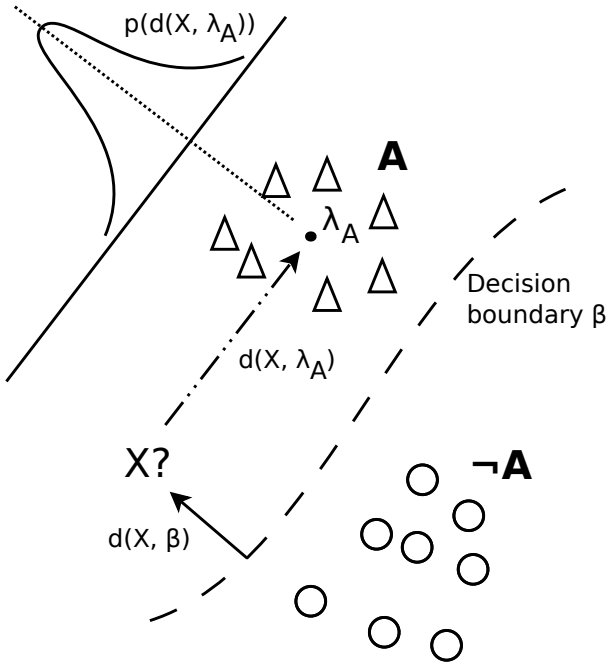


Figure 3: Separability vs. Prototypicality: For an unknown instance X, a large distance $d(X, \beta)$ from a margin β does not imply a short distance, $d(X, \lambda_A)$ from the prototype λ_A

class instances. This ability is usually achieved by evaluating the computed signed distance of an unknown sample to the decision boundary $d(X, \beta)$ which indicates on which side the instance X falls. However, by focusing on separation, an important aspect of pattern recognition is neglected: The phenomenon of *prototypicality* which concerns the similarity of an instance to the canonical class prototype, for instance, measured as the distance to the centroid or prototype of the class $d(X, \lambda_A)$. Quantitatively, prototypicality can be defined as $p(d(X, \lambda_A))$ and is also the underlying rationale for Bayesian classifiers, exploiting the high density of feature values around the mode of their distribution, as opposed to the SVM. It is important to realise that the prototypicality of instances directly affects the ease with which new training examples can be elicited from users in a continuously learning retrieval system. The degree of prototypicality of the hit list directly affects the gain factor in the feedback loop of the label harvesting system that is presented in Figure 2.

For a search and annotation tool of handwritten historical documents, separability and prototypicality need to be optimised simultaneously. It can be

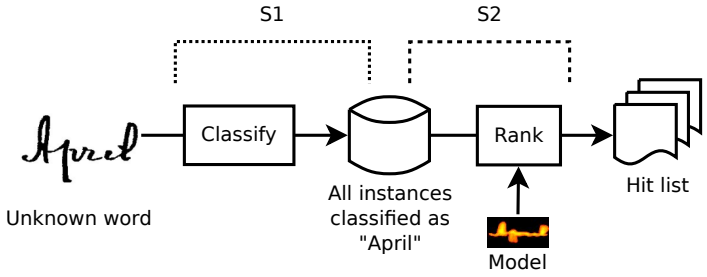


Figure 4: Schematic overview of the re-ranking process. The first stage (S1) shows that a word is classified first, and gathered together with other instances that have been classified the same. These instances are then ranked (S2), according to their prototypicality, to produce a ranked hit list.

argued that similar requirements play a role in general content-based image retrieval, too [9, 10]. However, most classifier methods optimise for one property, not both. The solution proposed in this study, is to combine classifiers in a two-stage process. The classifier that optimises separability is used in the first stage to divide the instances and produce the most likely class C for an unlabelled instance. The goal is to reduce the number of distractors for the second stage. More specifically, the set of distractors of an instance classified as C will be a considerable reduction of the set of all instances.

All instances labelled as C are then gathered for the second stage, where all instances are re-ranked or re-sorted with a secondary feature or method, one that optimises the ability to rank instances according to prototypicality. This ensures that if an instance is classified as class C in the first stage, but is an atypical result (such as the first few results in Figure 1, i.e., the speckles), the instance will end up at a later position in the hit list than other, more prototypical examples. Similar problems will occur if reject criteria need to be defined while using the SVM [11], or when there are very few negative examples to train from — for example, in a machine diagnostics problem[12]. For a schematic overview of the entire re-ranking process, see Figure 4.

The results from the SVM experiment in the introduction suggest that a larger number of distractors has a negative effect on retrieval precision. It should be noted that the experiments in this study are conducted in a laboratory setting, using only human labelled instances. In a real-world setting, the problem of distractors will even be worse: the problem space is then heavily populated with non-word images and other noise. For example, in Monk, over all collections there are 22×10^3 classes, with over 124×10^6 word images, including rejectable candidates and noise. These numbers indicate the massive

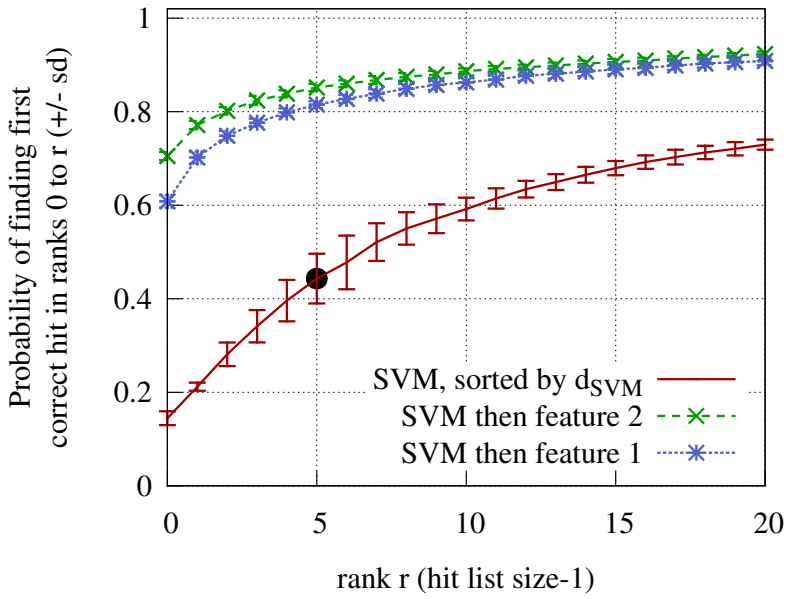


Figure 5: Probability of finding the first correct hit in ranks 0 to r for raw and ranked SVM output ($N_{\text{folds}} = 7$). The bars give the standard deviation, which are only clearly visible on the SVM, sorted by d_{SVM} results. Note the strong improvement due to secondary ranking for all ranks but especially for the top hit accuracy at $r = 0$. Feature 2 outperforms Feature 1 significantly. The circle is used as a reference point in the text.

size of the current experimental test bed. Instead of pre-cleaning the data, we assume a rigorous, machine-learning approach where as much of the problems are solved by the base classifier and not by the use of overly specific hand-coded preprocessing heuristics. That means that problematic patterns have to be labelled as well. In Monk, there are several classes that are indicated by a label starting with `@`, and can indicate whether this is, e.g., a table-line, speckles or other noise.

3. Methods

Figure 5 shows the probability of finding the first correct hit in the ranks 0 to r of the hit lists generated in the preliminary study from the introduction. It is apparent that the probability of finding the first correct hit in the first five ranks is roughly 45% (indicated by the circle in Figure 5), when using the SVM discriminant value for initial (tier 1) ranking. By reordering the images using a different feature, the performance can be improved, such that the first correct hit is found in the first five ranks 80% of the time (Figure 5, upper left).

This is hopeful, but this is not enough and the hit list still contains counter intuitive results in the top ranks. There are other ways of improving the tier-1 performance. For example, multiclass SVMs, using decision trees [13], could improve the classification accuracy before ranking, which seems to be beneficial, but it has the downside of requiring a large number of training instances for each of the more than 10^4 classes. Approaches like Gaussian mixture models (GMMs) or hidden Markov models (HMMs) can also improve the classification accuracy, but also require a large number of training examples. Benefits such as multi-peak distributions can be achieved with more simple techniques, such as (k-means) clustering. The *Monk* system is a continuous, '24/7' training system: Labels are continuously added or changed, and it would be too time consuming and require human monitoring to train and retrain SVM classifiers when the system is updated. Nearest-centroid classifiers, on the contrary, can be easily updated with new knowledge by just adding a new feature vector to the set of training samples and averaging the samples to get the centroid. Rather than constituting a simplistic old-fashioned method, nearest-neighbour approaches are at the core of important advances in computational linguistics [14] and image retrieval [15, 16]. The principle of central tendency leads to an intrinsic settling of centroid models as more examples are added. In case of multimodal distributions, occurring for example when there are multiple writing styles per class, clustering can be used to represent the class variants, e.g., by the k-means algorithm. Considering these multiple arguments, in this study, we will use a nearest-centroid classifier for the classification stage, instead of SVMs.

The choice of word-based image retrieval instead of character-based approaches is based, firstly, on the observation that in some historical document collections contractions and loops are used to suggest characters in order to speed up writing (see the marked images in Figure 6). This makes creating a mapping between letter identity and character shape non-trivial. Secondly, due to the large variety of scripts and languages, most character-based approaches would need to be fine-tuned for each script and language, leading to long projects to process new collections ("each book its PhD project"). Our goal is to collect huge numbers of labelled word images first over several collections and historical periods in order to develop character-based classifiers at a later stage, when necessary.

As discussed in the introduction, classification is performed by finding the class with the highest probability given the data. Since nearest neighbour classifiers are distance-based, the class with the highest probability is the class with the smallest distance to the instance:

$$\underset{i}{\operatorname{argmax}}\, P(C_i|X) = \underset{i}{\operatorname{argmin}}\, d(C_i|X) \tag{3}$$

Similarly, retrieval is performed by ranking all instances based on their distance to a class-model. Two features were experimentally chosen from a set of features to be used in the experiments. The exact implementation of both features is outside the scope of this article; different feature methods could be used instead without changing the actual re-ranking process. The first feature is based on the

Figure 6: This variety of styles and shapes in a realistic collection illustrates that 'optical character recognition' of handwriting, by some form of sliding window over a word, is only applicable to a small subset. Many patterns are abbreviations, linguistic contractions or suffer from deformed, 'suggested' characters (marked with asterisks). In the absence of character models, the total-word image on the contrary provides a rich and redundant pattern in all cases, and can be labelled easily by volunteers.

biologically inspired features introduced in [3], and the second is a more simple feature consisting of the normalised and scaled image. The dimensionality of the former feature is 4358, while the scaled image has a size of 100×50, yielding a comparable dimensionality of 5000. In both feature types, the feature vector consists of probability values, adding up to one.

Two methods of retrieval will be compared: 1) direct retrieval: ranking, in a single step, all instances from the test set with the distance of the image to the centroid of the target class, and 2) the two stage re-ranking method as described in the previous section: do recognition on all instances first, then for each class C rank its candidates. The re-ranking method can be done in four ways using the two features: recognition with either feature and ranking with either feature. All four combinations are used to study the effect of using a different, secondary feature in the re-rank phase.

There are a number of measures to be used for comparing recognition and

retrieval: (a) For recognition, we define top-1 recognition accuracy as: The probability that the nearest-centroid is of the correct class. For retrieval, the standard measures (b) precision and (c) recall will be considered, as well as (d) the *average edit distance* in the top-7 of each hit list.

Accuracy (a) is defined as the percentage correctly classified instances:

$$\text{Accuracy } = \frac{N_{correct}}{N_{total}} \tag{4}$$

with $N_{correct}$ is the total number of correctly classified instances (in the top-1), and N_{total} is the total number of instances. We are interested in accuracy because it can show which feature is a good choice for the first stage: features and methods with a high accuracy are well suited for classification.

Precision (b) is defined as the proportion of correctly retrieved instances of class C in a fixed hit list H, with target size n, and can be computed with

$$\text{Precision in top-}n = \frac{N_{correct}}{\min(n, |H|)} \tag{5}$$

where $N_{correct}$ is the number of instances with the correct label in the top-n and $|H|$ is the number of items in the hit list[1]. The minimum of n and $|H|$ is used because the hit list can be smaller than the target size of n items.

The recall measure (c) is defined as the proportion of instances of class C that can be found in the hit list; formally, it can be defined as

$$\text{Recall for class } C = \frac{N_{obtained}}{N_{targets}} \tag{6}$$

where $N_{obtained}$ is the number of instances retrieved with class C, and $N_{targets}$ is the total number of instances with class C in the given test set. The reported precision and recall are accumulated over all classes as proportions.

The concept of prototypicality cannot be seen in isolation from the application context. More specifically, users of a retrieval engine for historical handwritten words will have an evaluation of the quality of a hit list. In other words, $P(X_j|C)$ must reflect an underlying measure of similarity. In information retrieval, relevance feedback is used to estimate user appreciation[17]. Relevance feedback is outside the scope of this study, but to estimate the user appreciation, we use *average edit distance* as the fourth performance measure. The assumption is that if the text distance (in 'ASCII') between the query and the actual label of an instance is small, the hit list will be *intuitive*, meaning that it reflects the users measure of similarity well. The specific edit distance implemented in this study is the Levenshtein distance[18].

The data set is drawn from the historical document collection from the Dutch Queen's Office (see also [3]), or "Kabinet der Koningin" (KdK). The

[1] According to the Wikipedia article on precision and recall (`http://en.wikipedia.org/wiki/Precision_and_recall`, last accessed 23 January 2013), this is also called "precision at n" or "P@n"

Table 2: Top-1 accuracy ($N_{\text{folds}} = 7$)

Feature	N_{examples}							
	7-35		35-60		60-120		120+	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ
f1	**0.62**	$\pm$.02	**0.93**	$\pm$.01	**0.92**	$\pm$.01	**0.94**	$\pm$.00
f2	**0.62**	$\pm$.01	**0.86**	$\pm$.01	**0.87**	$\pm$.01	**0.93**	$\pm$.00

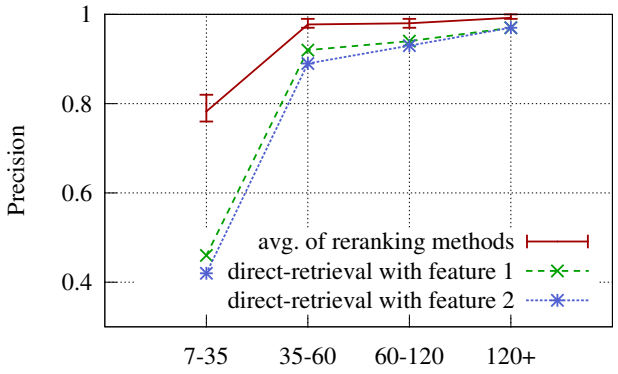
complete data set has over 13×10^3 classes. However, in order to do a 7-fold cross-validation experiment, only the 1404 classes with seven or more human labelled word instances will be considered. These classes will be divided into four categories, based on the number of instances: 7 up to 35 instances, 35 up to 60 instances, 60 up to 120 instances and 120 or more instances, similar to what has been done in [3]. This division is useful to compare performances when there are few labelled instances, a lot of labelled instances or in between. In total, there are more than 84×10^3 instances used. The experiments are performed on a cluster of eight Linux machines with 54 cores in total, connected to a 1.6 petabyte storage, of which the *Monk* system will use roughly 0.5 petabyte.

For each line strip, a number of word candidates are selected, based on the number and size of connected components. This means that the line is usually oversegmented, which leads to overlap between images. To avoid that multiple image renderings belonging to the same word instance end up in both the training and test set, the fold sets are compiled from exclusive page sets: fold $\equiv$ page number (mod N_{folds}), $N_{\text{folds}} = 7$. This has the additional, realistic benefit that trained words, which are written in a consistent style within one page, but inconsistently over the entire collection will not end up in the test set of a fold. Each fold holds 84 288 instances, of which the test set will hold $1/7^{\text{th}} \triangleq 12\,041$ instances on average.

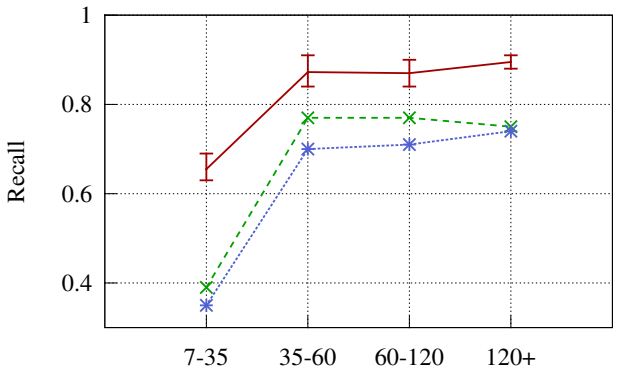
4. Results

We look at two types of comparisons: between re-rank methods (choice of features) and between average re-rank performance and direct retrieval (i.e., without re-ranking). Table 2 shows the top-1 recognition accuracy, averaged over all seven folds for both features. Feature 1 (f1) outperforms the second feature (f2), especially in the categories of 35-60 and 60-120 examples. Furthermore, the table shows that to accurately classify an instance, the nearest-centroid classifier needs around 35 training instances. Since feature 1 performs better than feature 2, it seems to be the best candidate for the classification step, as is confirmed below.

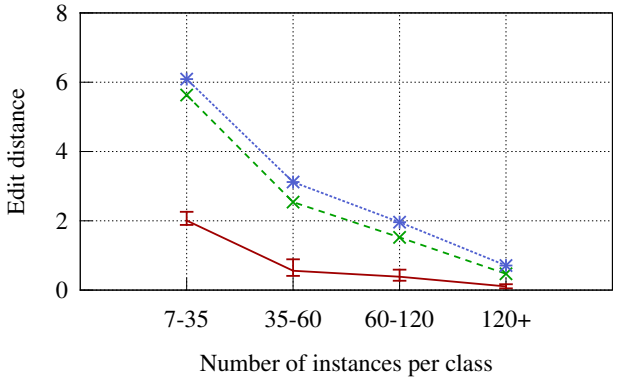
Figures 7a, 7b and 7c compare the average of the re-rank methods to the direct retrieval methods. The bars on the averages show the minimum and maximum value of the re-rank methods. These results show the gain in performance when using the re-ranking methods instead of direct retrieval. As was expected, reducing the number of distractors has a positive impact on performance.



(a) Precision performance in top-1



(b) Recall performance



(c) Average edit distance in top-7

Figure 7: Precision and recall performances (at $N \approx 1700$ and $\alpha = 0.01$, confidence is $\pm 3\%$) and average edit distance of re-rank vs. direct retrieval. The bars on the re-rank lines show the minimum and maximum performances of different feature configurations. All measures are averages over 7 folds.

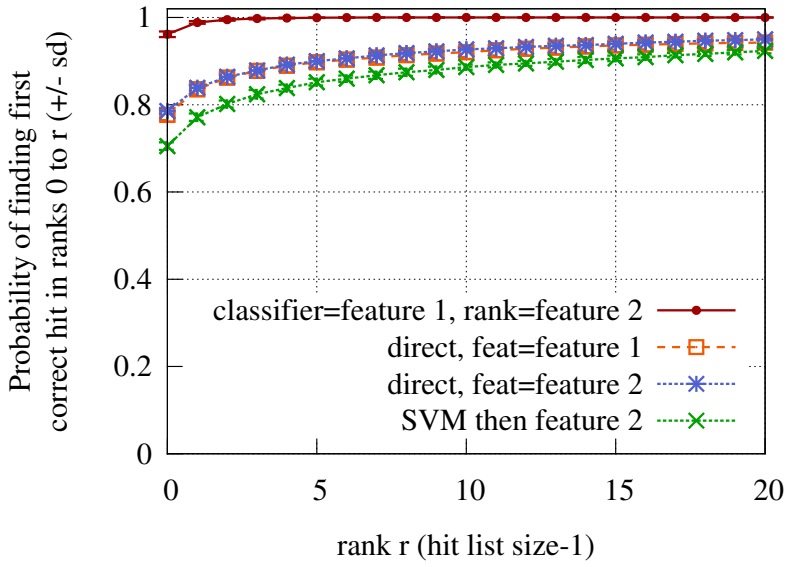


Figure 8: Probability of finding the first correct hit in ranks 0 to r for the re-rank method using feature 1 for classification and feature 2 for ranking, and the direct methods ($N_{\text{folds}} = 7$). The bars giving the standard deviations, are barely visible due to the large numbers of test instances in each fold (≈ 1700). The lines for both direct ranking methods are very close together and therefore not distinguishable from each other. The results show a considerable improvement in comparison to the raw, non-reranked results (Figure 5), especially for non-ranked SVM: The error at rank 0 is reduced from 29% to 4%, here.

Analogous to Figure 5, Figure 8 shows the probability of finding the first hit in ranks 0 to r for the re-rank method using feature 1 as the classification feature and feature 2 as the re-rank feature. The re-ranked method shows a considerable improvement from the direct ranking and the ranked SVM output (the best performance as reported in Figure 5). The probability of finding the first hit in the first four ranks even approaches 100%.

Table 3 and 4 show the precision (in the top-1) and recall figures. In general, these results show that re-ranking with a different feature can boost performance. The precision in top-7 for the re-rank methods is even higher than the precision in top-1 for the direct method, especially in the 7-35 category. Using feature 1 as a classification feature and feature 2 for ranking works best for this data collection, even getting a top-1 precision of 1.0 (i.e., 100%) with a standard deviation of 0 in the 120+ category.

Overall, the results show that all methods perform roughly the same when there are enough labelled samples (i.e., in the 120+ category).

Table 3: Precision results ($N_{\text{folds}} = 7$, $\sigma \leqslant 0.03$)

Method	N_{examples}			
	7-35	35-60	60-120	120+
Precision in top-1				
Direct, rank with f2	0.42	0.89	0.93	0.97
Direct, rank with f1	0.46	0.92	0.94	0.97
Re-rank, classify with f2, rank with f2	0.76	0.97	0.98	0.99
Re-rank, classify with f2, rank with f1	0.76	0.97	0.98	0.99
Re-rank, classify with f1, rank with f1	0.79	0.98	0.97	0.99
Re-rank, classify with f1, rank with f2	**0.82**	**0.99**	**0.99**	**1.00**
Precision in top-7				
Direct, rank with f2	0.14	0.52	0.71	0.90
Direct, rank with f1	0.15	0.57	0.75	0.91
Re-rank, classify with f2, rank with f2	0.64	0.87	0.91	0.97
Re-rank, classify with f2, rank with f1	0.68	0.91	0.94	0.98
Re-rank, classify with f1, rank with f1	0.69	0.93	0.94	0.97
Re-rank, classify with f1, rank with f2	0.69	0.93	**0.95**	**0.99**

Table 4: Recall results ($N_{\text{folds}} = 7$, $\sigma \leqslant 0.03$)

Method	N_{examples}			
	7-35	35-60	60-120	120+
Direct, rank with f2	0.35	0.70	0.71	0.74
Direct, rank with f1	0.39	0.77	0.77	0.75
Re-rank, classify with f2, rank with f2	0.63	0.84	0.84	0.88
Re-rank, classify with f2, rank with f1	0.63	0.84	0.85	0.89
Re-rank, classify with f1, rank with f1	0.67	0.90	0.89	0.90
Re-rank, classify with f1, rank with f2	**0.69**	**0.91**	**0.90**	**0.91**

5. Conclusions

In the design of a large scale retrieval engine for historical handwritten manuscripts it was observed that classifier accuracy is not a good predictor of retrieval precision. Very low precision performances occurred on good classifiers when using a realistic number of distractors. In retrospect, the choice of using the signed distance d_{SVM} from the margin for ranking was evidently suboptimal, but it elucidated two separate functions to be performed: 1) data reduction by optimal *separation* and 2) ranking instances in terms of their *prototypicality* with respect to their class.

The re-ranking method has two main advantages: the focus on both separability and prototypicality increases the probability that the top of a hit list is more similar to the user's expectation than otherwise. Secondly, the reduction of distractors lowers the number of noisy instances in a hit list and is advantageous in terms of processing demands. As the results presented in the previous section show, reducing the number of distractors in a retrieval experiment improves precision and decreases average edit distance in the hit list, which we assume will increase the user appreciation of hit lists. We think that a simultaneous solution of separability and prototypicality will suffer from a performance reduction that is typical of Pareto curves in multi-objective optimisation, but

this is a matter of future research. To investigate whether we can optimise both separability and prototypicality in the SVM paradigm, we performed some preliminary tests. These tests show that weighing the discriminant value d_{SVM} with the distance to the centroid of positive examples $e^{-d(\lambda,X)}$ does not have positive effects on precision. Future research will look into other multi-objective approaches involving both separability and prototypicality.

It appeared to be beneficial for retrieval performance to use different features in the separate stages. While the processing order is fixed — separation first, ranking second — the selection of optimal features and machine learning algorithms will depend on the material. In the KdK data set, precision benefited the most by using a strong, robust feature for recognition first, and a secondary feature with a strong image-based component that works well on collections where words are written fairly consistently. On data sets where the writing varies a lot within a class, other features or classifier methods may prove to be more advantageous, including (k-means) clustering to capture the different writing styles. A system like Monk will have several tool libraries and approaches for diverse material. The optimality of the parameters for a complete processing pipeline depends on the ink deposition process, writing style and physical material. Improving the recognition accuracy using linguistic models and contextual information is difficult due to the nature of the material. While linguistic models offer improved transcription performances for contemporary texts, previous efforts of using contextual information[19, 20] proved not to be robust enough for use in our system because there are no useful corpora available for the document collections we deal with. This is due to the abundance of abbreviations, contractions and named entities that are not found in corpora of contemporary text. Furthermore, in certain document collections, several languages are used, sometimes even in the same paragraph. Corpora for transcription systems for contemporary texts usually contain millions of words gathered from various sources[21, 22], which we can not provide for the bootstrapping of handwriting recognition for the document collections in Monk.

When a class has enough instances (i.e., the 120+ category), choice of feature does not seem to have much effect on retrieval performance. On the other hand, reducing the number of distractors by a two-step approach is still beneficial. In the bootstrapping phase of a retrieval system (i.e., the category of 7-35 training examples), the choice of feature does have a big impact. Even small accuracy performance increases have large consequences in this stage, helping the user to label new instances with little effort (since *Monk* presents hit lists in its web-based labelling interface).

The methods presented in this paper can use all kinds of classifiers. Currently, nearest-centroid classifiers are used due to the nature of '24/7' learning, where new labels are being added frequently. It would be cumbersome to retrain classifiers such as SVMs every time a new label was added. The SVM has one benefit in the bootstrap phase: its recognition accuracy is better than the performance of a nearest neighbour classifier. However, the 7-35 category in this experiment has the most classes by far, which would be very inconvenient for the training of tens of thousands multi-class SVMs. This touches

on the fundamental difference between SVMs and Bayesian classifiers. While Bayesian classifiers, including nearest centroid classification, will incorporate the retention of the degree of prototypicality in the "1 out of N" choice itself (i.e., $p(d(X, \lambda))$), a tree of SVMs capitalizes on separability, only.

The *Monk* project has a large number of collections with different script types: 15^{th} (mixed languages, frequent use of word contractions) and late 19^{th} century texts (cursive with a lot of abbreviations and variation), Qumran scrolls (isolated characters), captain's logs (cursive) and even Thai[23] and Bangla[24] texts. The different shapes and writing styles have different requirements of the features; For each script, features will be selected to optimise both separability and prototypicality.

Summarising, we found that the assumption that a good recognizer will also be good at ranking is not intrinsically tenable. Two requirements need to be fulfilled. First, a method (feature and classifier) is selected based on its ability to separate class instances from non-class instances. Subsequently, a method (feature and classifier) is selected on the basis of its ability to rank instances according to prototypicality, such that the final ranking is similar to the users expectation. This stepwise approach yielded very substantial improvements in precision, substantial improvements in recall as well as a substantial reduction of the edit distance, a measure of word-match intuitiveness. Finally, the insight that separation and ranking of instances both need to be optimised may have a broad applicability beyond handwriting recognition.

References

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, 2001.

[2] Bunke, H., Recognition of cursive Roman handwriting: past, present and future, in: Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on, IEEE, 2003, pp. 448–459.

[3] T. van der Zant, L. Schomaker, K. Haak, Handwritten-word spotting using biologically inspired features, Pattern Analysis and Machine Intelligence, IEEE Transactions on 30 (2008) 1945–1957.

[4] U. Marti, H. Bunke, Handwritten sentence recognition, in: Proceedings of the 15th International Conference on Pattern Recognition, volume 3, IEEE, 2000, pp. 463–466.

[5] T. Artières, S. Marukatat, P. Gallinari, Online handwritten shape recognition using segmental hidden Markov models, Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (2007) 205–217.

[6] V. Vapnik, Estimation of Dependencies Based on Empirical Data, Springer-Verlag, New York, 1982.

[7] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.

[8] T. van der Zant, L. Schomaker, S. Zinger, H. van Schie, Where are the search engines for handwritten documents?, Interdisciplinary Science Reviews, 34 2 (2009) 224–235.

[9] R. Datta, D. Joshi, J. Li, J. Z. Wang, Image retrieval: Ideas, influences, and trends of the new age, ACM Comput. Surv. 40 (2008) 5:1–5:60.

[10] L. Schomaker, E. de Leau, L. Vuurpijl, Using pen-based outlines for object-based annotation and image-based queries, in: Proceedings of the Third International Conference on Visual Information and Information Systems, VISUAL '99, Springer-Verlag, London, UK, UK, 1999, pp. 585–592.

[11] H. Mouchère, Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs: Application à la reconnaissance de l'écriture manuscrite en-ligne, Ph.D. thesis, l'Institut National des Sciences Appliquées de Rennes, 2007.

[12] D. Tax, One-class classification, Ph.D. thesis, Technische Universiteit Delft, 2001.

[13] F. Takahashi, S. Abe, Decision-tree-based multiclass support vector machines, in: Proceedings of the 9th International Conference on Neural Information Processing, volume 3, IEEE, 2002, pp. 1418–1422.

[14] W. Daelemans, A. van den Bosch, Memory-based language processing, Cambridge Univ Pr, 2005.

[15] G. Giacinto, A nearest-neighbor approach to relevance feedback in content based image retrieval, in: Proceedings of the 6th ACM international conference on Image and video retrieval, ACM, 2007, pp. 456–463.

[16] H. Jégou, M. Douze, C. Schmid, Improving bag-of-features for large scale image search, International Journal of Computer Vision 87 (2010) 316–336.

[17] G. Salton, C. Buckley, Improving retrieval performance by relevance feedback, Readings in information retrieval 24 (1997) 5.

[18] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, volume 10, 1966, pp. 707–710.

[19] M. P. Ritsema van Eck, L. Schomaker, Formal semantic modeling for human and machine-based decoding of medieval manuscripts., in: Digital Humanities, Hamburg, 2012. URL: `http://www.dh2012.uni-hamburg.de/conference/programme/abstracts/formal-semantic-modeling-for-human-and-machine-based-decoding-of-medieval-manuscripts/`.

[20] S. Zinger, J. Nerbonne, L. Schomaker, Text-image alignment for historical handwritten documents, in: IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, 2009, pp. 724703–724703.

[21] M. Zimmermann, H. Bunke, N-gram language models for offline handwritten text recognition, in: Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on, IEEE, 2004, pp. 203–208.

[22] J. Devlin, M. Kamali, K. Subramanian, R. Prasad, P. Natarajan, Statistical machine translation as a language model for handwriting recognition, in: Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on, IEEE, 2012, pp. 291–296.

[23] O. Surinta, L. Schomaker, M. Wiering, Handwritten character classification using the hotspot feature extraction technique, in: Proceedings of the First International Conference on Pattern Recognition Applications and Methods, 2012, 2012, pp. 261–264.

[24] T. Bhowmik, J. van Oosten, L. Schomaker, Segmental K-means learning with mixture distribution for HMM based handwriting recognition, Pattern Recognition and Machine Intelligence (2011) 432–439.

Lambert Schomaker is professor of artificial intelligence at the University of Groningen, The Netherlands. He has produced over 140 publications, predominantly in pattern recognition, is member of IEEE and IAPR, and received the IBM Faculty Awards (2011,2012) for the Monk word retrieval system in historical manuscript collections using high-performance computing.

Jean-Paul van Oosten received the MSc degree cum laude in artificial intelligence from the University of Groningen, The Netherlands, in 2010. He received the IAPR Best Paper award at the ICFHR 2012 conference and is currently a PhD student at the Institute of Artificial Intelligence and Cognitive Engineering in Groningen.